REMARKS

In the Office action mailed on May 7, 2004, claims 1–16 are rejected. Claims 1 and 9 are provisionally rejected based on obviousness-type double patenting over claims 1 and 18 of copending Application No. 09/737,108 (*Copending Application*). Claims 1–16 are rejected under 35 U.S.C. 102 over U.S. Patent Publication No. 2001/0036834 to Das et al. ("*Das*"). In addition, the Action objects to the drawings. Applicant traverses the rejections and objections, amends the drawings and specification, and submits the enclosed Terminal Disclaimer and Affidavit of Gregory D. Troxel under 37 CFR 1.131.

Applicants submit that the amendments to the specification and drawing satisfy the objection and request that the objection be withdrawn.

Applicant traverses the double patenting rejection. However, in order to move prosecution forward, Applicant submits the enclosed Terminal Disclaimer. The *Copending Application* and the present application are commonly owned and have the same inventorship.

The Claimed Invention Was Invented Prior to the Priority Date of *Das*:

The Action rejects claims 1–16 under 35 USC § 102(b) over U.S. Patent Publication No. 2001/0036834, filed on March 2, 2001. The application was filed on December 14, 2000, prior to the filing of *Das*. Therefore, *Das* cannot be used as a § 102(b) reference in relation to the application.

The Action also cites the language of 35 USC 102(e) as the basis for the § 102 rejection of the claims. Applicants traverse the § 102(e) rejection without addressing the merits of the Action's assertion that *Das* discloses each and every element of the rejected claims. For a published application to apply as a reference under § 102(e), the published application must be entitled to a filing date that is "before the invention by the applicant for patent." Applicants submit that they invented the claimed subject matter prior to the filing of the earliest priority document to which *Das* claims the benefit. *Das* claims priority to U.S. Provisional Patent Application No. 60/186,910, filed March 3, 2000. Applicants submit that the enclosed Affidavit of Gregory D. Troxel under 37 CFR 1.131 and software source code attached thereto demonstrate that the subject matter of independent claims 1 and 9 was both conceived of and reduced to practice prior to March 3, 2000. In light of the above, Applicants request reconsideration of the claims and withdrawal of the §102 rejections.

Applicant believes no fee is due with this response other than as reflected on the enclosed Fee Transmittal. However, if a fee is due, please charge our Deposit Account No. 18-1945, under Order No. BBNT-P01-107 from which the undersigned is authorized to draw.

Dated: September 23, 2004

Respectfully submitted,

By [illegible]
Edward A. Gordon
Registration No.: 54,130
ROPES & GRAY LLP
One International Place
Boston, Massachusetts 02110-2624
(617) 951-7000
(617) 951-7050 (Fax)
Attorneys/Agents For Applicant

I hereby certify that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as First Class Mail, in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date shown below.

Dated: 9/23/04 Signature: Joanne Ryan
Joanne Ryan

Docket No.: BBNT-P01-107
(PATENT)

D TRADEMARK OFFICE

OIPE
SEP 2 7 2004
PATENT & TRADEMARK OFFICE
JC181

In re Patent Application of:
Troxel et al.

Application No.: 09/736807

Confirmation No.: 6622

Filed: December 14, 2000

Art Unit: 2666

For: DELIVERING MESSAGES TO A NODE AT A FOREIGN NETWORK

Examiner: W. Eugene

AFFIDAVIT UNDER 37 C.F.R. 1.131

I, Gregory D. Troxel, do hereby declare as follows:

1. I am one of the inventors of the subject matter claimed and described in the above referenced patent application.
2. I, with my coinventor, conceived of and reduced to practice the subject matter of the pending claims prior to March 3, 2000, the asserted priority date of the subject matter described in U.S. Patent Application 09/798,468 to Das et al.
3. The attached code, entered into the code archive system of Assignee, BBNT Solutions LLC, prior to March 3, 2000, illustrates the relevant portions of a computer program as described in claims 9–16 that carries out the methods described in claims 1–8. The code illustrates one embodiment of the invention and is not intended to narrow the scope of the claims. The code has not been altered from its archived state other than to remove the comments contained therein.
4. In particular, the attached code includes the following functions and procedures:

a) process_localBindingRequest, running on a first router, receives a network layer address of a first node (having a different home subnetwork than the first router) on the first router's link. The procedure installs a local binding between the first router and the first node.

b) process_mn_notification, running, on the first node, observes the network layer addresses included in the headers of processed data packets. If the procedure detects packets intended for a second node (a first remote node) on a second non-home-subnetwork link, the procedure initiates the sending of the network layer address of the first node and the network layer address of the first router to the first remote node as a remote binding request.

c) process_remoteBindingRequest, running on the first remote node, receives the remote binding request sent from the first node and installs a remote binding on the first remote node between the first remote node and the first node. The installation of a remote binding places an entry in the first remote node's kernel routing table linking the network layer address of the first node to a tunnel to the network layer address of first router.

d) in_gif.c, running on the first router, examines received packets by calling in_gif_input to determine whether a received packet was received via a tunnel, intended to be forwarded to the first node. If in_gif_input detects such a packet, in_gif.c detunnels the packet using the m_adj function and forwards the packet to the first node using ip_forward based upon the link layer address stored in the first router's kernel routing table, which was installed by process_localBindingRequest.

5. I assert that all statements made of my own knowledge are true, and that all

statements made on information and belief are believed to be true. I also understand that willful false statements and the like are punishable by fine or imprisonment, or both (18 U.S.C. § 1001) and may jeopardize the validity of the application or any patent issuing thereon.

September 22, 2007

Dated

[illegible]

Gregory D. Troxel

REDACTED

process_localBindingRequest

```
process_localBindingRequest (struct packet packet) {

   struct interface_info *ip = packet.interface;
   struct mip_lbnd_request *lbndreq =
     (struct mip_lbnd_request *)&packet.raw->m;

   struct binding *bp;
   struct binding *flooded;
   struct mip_lbnd_reply *lbndrply;
   struct msg_info msginfo;


   msginfo.local_if = ip;
   msginfo.remote_if = setup_peer_info(packet.sender,
                             packet.sender_haddr);


   GET_TIME(&cur_time);

   if(lbndreq->lifetime == 0) {
     debug("   Got a local binding cancel msg from: %s",
           inet_ntoa(lbndreq->sender));
     cancel_binding(GRANTED_LOCAL_BINDING, packet.sender,
                    CANCEL_EXPIRE);
     return;
   }

   debug("   Got a local binding request from: %s",
       inet_ntoa(lbndreq->sender));


   make_localBindingReply(ip, lbndreq, &msginfo.pkt);
   lbndrply = (struct mip_lbnd_reply *)&msginfo.pkt->m;
   if(!lbndrply->errors) {
     bp = grant_localBinding(&packet, lbndrply->lifetime);
     install_binding(bp);
   }
   send_localBindingReply((void*)&msginfo);

   free_amis_packet(msginfo.pkt, "msginfo_msg");
   free_peerinfo(msginfo.remote_if, "peer");
}
```

REDACTED

mn_notification

```
void process_mn_notification(struct in_addr *dst_addr,
                              u_int16_t count)
{
    struct in_addr all_brdcst = {0}, our_brdcst = {0};
    struct in_addr our_subnets = {0}, our_netmask = {0};
    struct in_addr dst_net = {0};
    struct in_addr home_addr;
    struct iaddr dstaddr;
    struct interface_info *ip;
    struct binding *lb, *rb;


    if(!accepting_notifications)
      return;

    for(ip = interfaces; ip; ip=ip->next) {
      if(ntohl(ip->primary_address.s_addr) ==
         ntohl(dst_addr->s_addr))
      return;
    }

    if(ntohl(dst_addr->s_addr) == ntohl(home_agent.s_addr))
      return;

    if(IN_MULTICAST(ntohl(dst_addr->s_addr)))
      return;

    inet_aton("255.255.255.255", &all_brdcst);
    inet_aton(piaddr(notif_ip->subnet->broadcast), &our_brdcst);
    if((ntohl(dst_addr->s_addr) == ntohl(all_brdcst.s_addr)) ||
       (ntohl(dst_addr->s_addr) == ntohl(our_brdcst.s_addr)))
      return;

    if(me.protocol_subnets.len != 0) {
      inet_aton(piaddr(me.protocol_subnets), &our_subnets);
      inet_aton(piaddr(me.protocol_netmask), &our_netmask);
      dst_net.s_addr = (dst_addr->s_addr & our_netmask.s_addr);
      if(ntohl(dst_net.s_addr) != ntohl(our_subnets.s_addr))
          return;
    }

    debug("Notification for dst: %s", inet_ntoa(*dst_addr));

    dstaddr.len = sizeof(dst_addr);
    memcpy(&dstaddr.iabuf, dst_addr, dstaddr.len);
    rb = find_binding(ACQUIRED_REMOTE_BINDING, dstaddr);
    if(rb) {
      GET_TIME(&cur_time);
      rb->ai->last_notification = cur_time;
    }
    else
      state_acquire_addrb(notif_ip, dstaddr);
}
```

REDACTED

process_remoteBindingRequest

```
void process_remoteBindingRequest (struct packet packet) {

    struct interface_info *ip = packet.interface;
    struct mip_rbnd_request *rbndreq =
      (struct mip_rbnd_request *)&packet.raw->m;

    struct binding *bp;
    struct binding *flooded;
    struct mip_rbnd_reply *rbndrply;
    struct msg_info msginfo;

    msginfo.local_if = ip;
    msginfo.remote_if = setup_peer_info(packet.sender,
                              packet.sender_haddr);


    GET_TIME(&cur_time);
    if(rbndreq->lifetime == 0) {
      debug("   Got a remote binding cancel msg from: %s",
            inet_ntoa(rbndreq->sender));
      cancel_binding(GRANTED_REMOTE_BINDING, packet.sender,
                     CANCEL_EXPIRE);
      return;
    }

    debug("   Got a remote binding request from: %s",
        inet_ntoa(rbndreq->sender));

    make_remoteBindingReply(ip, rbndreq, &msginfo.pkt);
    rbndrply = (struct mip_rbnd_reply *)&msginfo.pkt->m;
    if(!rbndrply->errors) {
      bp = grant_remoteBinding(&packet, rbndrply->lifetime);
      install_binding(bp);
    }
    send_remoteBindingReply((void*)&msginfo);

    free_amis_packet(msginfo.pkt, "msginfo_msg");
    free_peerinfo(msginfo.remote_if, "peer");
}
```

REDACTED

in_gif.c

```
/*
 * Copyright (C) 1995, 1996, 1997, and 1998 WIDE Project.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. Neither the name of the project nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE PROJECT AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE PROJECT OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */


#ifdef __FreeBSD__
#include "opt_mrouting.h"
#if __FreeBSD__ >= 3
#include "opt_inet.h"
#endif
#endif

#include <sys/param.h>
#include <sys/systm.h>
#include <sys/socket.h>
#include <sys/sockio.h>
#include <sys/mbuf.h>
#include <sys/errno.h>
#if !defined(__FreeBSD__) || __FreeBSD__ < 3
#include <sys/ioctl.h>
#endif
#include <sys/protosw.h>

#include <net/if.h>
#include <net/route.h>
#include <net/if_gif.h>

#include <netinet/in.h>
#include <netinet/in_systm.h>
```

REDACTED

```
#include <netinet/ip.h>
#include <netinet/ip_var.h>
#include <netinet/in_gif.h>
#include <netinet/ip_ecn.h>

#ifdef INET6
#include <netinet/ip6.h>
#endif

#ifdef MROUTING
#include <netinet/ip_mroute.h>
#endif

#include <net/if_gif.h>


#ifdef __NetBSD__
#include <machine/stdarg.h>
#endif

int
in_gif_output(ifp, family, m, rt)
      struct ifnet      *ifp;
      int          family;
      struct mbuf *m;
      struct rtentry *rt;
{
      register struct gif_softc *sc = (struct gif_softc*)ifp;
      struct sockaddr_in *dst = (struct sockaddr_in *)&sc->gif_ro.ro_dst;
      struct sockaddr_in *sin_src = (struct sockaddr_in *)sc->gif_psrc;
      struct sockaddr_in *sin_dst = (struct sockaddr_in *)sc->gif_pdst;
      struct ip iphdr;
      int proto, error;
      u_int8_t tos;

      if (sin_src == NULL || sin_dst == NULL ||
          sin_src->sin_family != AF_INET ||
          sin_dst->sin_family != AF_INET) {
            m_freem(m);
            return EAFNOSUPPORT;
      }

      switch (family) {
#ifdef INET
      case AF_INET:
          {
            struct ip *ip;

            proto = IPPROTO_IPV4;
            if (m->m_len < sizeof(*ip)) {
                  m = m_pullup(m, sizeof(*ip));
                  if (!m)
                        return ENOBUFS;
            }
            ip = mtod(m, struct ip *);
            tos = ip->ip_tos;
            break;
```

REDACTED

```
		}
#endif
#ifdef INET6
	case AF_INET6:
		{
		 struct ip6_hdr *ip6;
		 proto = IPPROTO_IPV6;
		 if (m->m_len < sizeof(*ip6)) {
			m = m_pullup(m, sizeof(*ip6));
			if (!m)
				return ENOBUFS;
		 }
		 ip6 = mtod(m, struct ip6_hdr *);
		 tos = (ntohl(ip6->ip6_flow) >> 20) & 0xff;
		 break;
		}
#endif
	default:
#ifdef DIAGNOSTIC
		printf("in_gif_output: warning: unknown family %d passed\n",
			family);
#endif
		m_freem(m);
		return EAFNOSUPPORT;
	}

	bzero(&iphdr, sizeof(iphdr));
	iphdr.ip_src = sin_src->sin_addr;
	if (ifp->if_flags & IFF_LINK0) {

		if (sin_dst->sin_addr.s_addr != INADDR_ANY)
			iphdr.ip_dst = sin_dst->sin_addr;
		else if (rt) {
			iphdr.ip_dst = ((struct sockaddr_in *)
					(rt->rt_gateway))->sin_addr;
		} else {
			m_freem(m);
			return ENETUNREACH;
		}
	} else {
		if (sin_dst->sin_addr.s_addr != INADDR_ANY)
			iphdr.ip_dst = sin_dst->sin_addr;
		else {
			m_freem(m);
			return ENETUNREACH;
		}
	}
	iphdr.ip_p = proto;

	iphdr.ip_ttl = GIF_TTL;
	iphdr.ip_len = m->m_pkthdr.len + sizeof(struct ip);
	if (ifp->if_flags & IFF_LINK1)
		ip_ecn_ingress(ECN_ALLOWED, &iphdr.ip_tos, &tos);


	M_PREPEND(m, sizeof(struct ip), M_DONTWAIT);
	if (m && m->m_len < sizeof(struct ip))
```

REDACTED

```
		m = m_pullup(m, sizeof(struct ip));
	if (m == NULL) {
		printf("ENOBUFS in in_gif_output %d\n", __LINE__);
		return ENOBUFS;
	}

	*(mtod(m, struct ip *)) = iphdr;

	if (dst->sin_family != sin_dst->sin_family ||
	    dst->sin_addr.s_addr != sin_dst->sin_addr.s_addr) {

		dst->sin_family = sin_dst->sin_family;
		dst->sin_len = sizeof(struct sockaddr_in);
		dst->sin_addr = sin_dst->sin_addr;
		if (sc->gif_ro.ro_rt) {
			RTFREE(sc->gif_ro.ro_rt);
			sc->gif_ro.ro_rt = NULL;
		}
#if 0
		sc->gif_if.if_mtu = GIF_MTU;
#endif
	}

	if (sc->gif_ro.ro_rt == NULL) {
		rtalloc(&sc->gif_ro);
		if (sc->gif_ro.ro_rt == NULL) {
			m_freem(m);
			return ENETUNREACH;
		}
#if 0
		ifp->if_mtu = sc->gif_ro.ro_rt->rt_ifp->if_mtu
			- sizeof(struct ip);
#endif
	}

#ifdef IPSEC
	m->m_pkthdr.rcvif = NULL;
#endif
	error = ip_output(m, 0, &sc->gif_ro, 0, 0);
	return(error);
}

void
#ifndef __NetBSD__
in_gif_input(m, off, proto)
	struct mbuf *m;
	int off;
	int proto;
{
#else
#if __STDC__
in_gif_input(struct mbuf *m, ...)
#else
in_gif_input(m, va_alist)
	struct mbuf *m;
	va_dcl
#endif
```

REDACTED

```
{
      int off, proto;
#endif
      struct gif_softc *sc;
      struct ifnet *gifp = NULL;
      struct ip *ip;
      int i, af;
#ifdef __NetBSD__
      va_list ap;
#endif
      u_int8_t otos;

#ifdef __NetBSD__
      va_start(ap, m);
      off = va_arg(ap, int);
      proto = va_arg(ap, int);
      va_end(ap);
#endif

      ip = mtod(m, struct ip *);


#define     satosin(sa) ((struct sockaddr_in *)(sa))
      for (i = 0, sc = gif; i < ngif; i++, sc++) {
            if (sc->gif_psrc == NULL
             || sc->gif_pdst == NULL
             || sc->gif_psrc->sa_family != AF_INET
             || sc->gif_pdst->sa_family != AF_INET) {
                  continue;
            }

            if ((sc->gif_if.if_flags & IFF_LINK0)
             && satosin(sc->gif_psrc)->sin_addr.s_addr == ip->ip_dst.s_addr
             && satosin(sc->gif_pdst)->sin_addr.s_addr == INADDR_ANY) {
                  gifp = &sc->gif_if;
                  continue;
            }

            if (satosin(sc->gif_psrc)->sin_addr.s_addr == ip->ip_dst.s_addr
             && satosin(sc->gif_pdst)->sin_addr.s_addr == ip->ip_src.s_addr)
            {
                  gifp = &sc->gif_if;
                  break;
            }
      }

      if (gifp == NULL) {
#ifdef MROUTING

            if (proto == IPPROTO_IPV4) {
                  ipip_input(m, off, proto);
                  return;
            }
#endif
            m_freem(m);
            ipstat.ips_nogif++;
            return;
```

```
	}

	otos = ip->ip_tos;
	m_adj(m, off);

	switch (proto) {
#ifdef INET
	case IPPROTO_IPV4:
	    {
		struct ip *ip;
		af = AF_INET;
		if (m->m_len < sizeof(*ip)) {
			m = m_pullup(m, sizeof(*ip));
			if (!m)
				return;
		}
		ip = mtod(m, struct ip *);
		if (gifp->if_flags & IFF_LINK1)
			ip_ecn_egress(ECN_ALLOWED, &otos, &ip->ip_tos);
		break;
	    }
#endif
#ifdef INET6
	case IPPROTO_IPV6:
	    {
		struct ip6_hdr *ip6;
		u_int8_t itos;
		af = AF_INET6;
		if (m->m_len < sizeof(*ip6)) {
			m = m_pullup(m, sizeof(*ip6));
			if (!m)
				return;
		}
		ip6 = mtod(m, struct ip6_hdr *);
		itos = (ntohl(ip6->ip6_flow) >> 20) & 0xff;
		if (gifp->if_flags & IFF_LINK1)
			ip_ecn_egress(ECN_ALLOWED, &otos, &itos);
		ip6->ip6_flow &= ~htonl(0xff << 20);
		ip6->ip6_flow |= htonl((u_int32_t)itos << 20);
		break;
	    }
#endif
	default:
		ipstat.ips_nogif++;
		m_freem(m);
		return;
	}
	gif_input(m, af, gifp);
	return;
}
```

REDACTED

acquire_remoteBinding

```
struct binding *acquire_remoteBinding(struct interface_info *ip,
                              struct peer_info *peer)
{
    struct binding *bp;
    struct mip_rbnd_request *rbndreq;
    struct msg_info *msginfo = new_msginfo("message");

    assert(!(find_binding(ACQUIRED_REMOTE_BINDING, peer->addr)));

    msginfo->local_if = ip;
    msginfo->remote_if = peer;

    bp = create_remoteBinding(ip, peer);
    add_binding(bp);

    if(addr_eq(ip->subnet->home_agent, peer->addr)) {
      bp->characteristics |= PRIMARY;
      bp->ai->cb = remote_binding_callback;
    }
    else {
      bp->characteristics |= ADDITIONAL;
      bp->ai->add_remote_state |= S_ACQUIRE_ADDRB;
      bp->ai->cb = add_remote_binding_callback;
    }

    make_remoteBindingRequest(ip, peer, &msginfo->pkt);
    bp->ai->msginfo = msginfo;

    rbndreq = (struct mip_rbnd_request *)&msginfo->pkt->m;
    bp->lifetime = rbndreq->lifetime;
    set_bindingTimers(bp);

    GET_TIME(&cur_time);
    bp->ai->binding_attempted = cur_time;
    bp->characteristics |= IN_PROCESS;
    send_remoteBindingRequest((void *)msginfo);

    return(bp);
}
```

REDACTED

```
void state_acquire_addrb(struct interface_info *ip,
                   struct iaddr peer_addr)
{
    struct peer_info *peer;
    struct hardware peer_haddr = {0};

    peer = setup_peer_info(peer_addr,
                     peer_haddr);

    acquire_remoteBinding(ip, peer);

}
```

AMENDMENTS TO THE DRAWINGS

The attached sheet(s) of drawings includes changes to Fig. 24.

Attachment: Replacement sheet